MidTerm Exam II, Fall 2018 | November 6th | Exam time: 17:00-18:30

Student's [redacted] ID [redacted] Section: 11:00 AM

23.5/30

Problem 1 (2 points) 2/2

Compare dynamic programming and standard recursion by filling out the table below:

Algorithm	*Top-down or bottom-up?*	*Solve the same subproblem once?*
Dynamic Programming	Top-down ✓	once. ✓
Standard recursion	bottom-UP ✓	more the once

Problem 2 (9 points) 8/9

Let X and Y be two strings such that X= "recursion" and Y= "election"

We define X_i and Y_j as two prefixes of X and Y of length i and j, respectively.

We use a matrix C to store the optimal solutions of the subproblems. Let $C[i, j]$ be the length of Longest Common Subsequence (LCS) of X_i and Y_j.

(a) For X_i and Y_j, what must be true about $C[i, j]$?

$$c[i,j] = \begin{cases} c[i-1, j-1]+1 , & x_i = Y_j \\ \max\{c[i-1,j], c[i,j-1]\} , & \text{otherwise} \end{cases}$$

It's $O(m \cdot n)$

(b) Compute the length of an LCS of X and Y by filling out the matrix C.

* The table is in (c).

(c) What is LCS of *X* and *Y*? Explain how you can obtain the LCS from the table **C**.

		Y_j	e 1	l 2	e 3	c 4	t 5	i 6	o 7	n 8
	X_i	0	0	0	0	0	0	0	0	0
1	r	0	0	0	0	0	0	0	0	0
2	e	0	1	1	1	1	1	1	1	1
3	c	0	1	1	1	2	2	2	2	2
4	u	0	1	1	1	2	2	2	2	2
5	r	0	1 ←	1 ←	1 ↖	2	2	2	2	2
6	s	0	1	1	1	2 ←	2 ↖	2	2	2
7	i	0	1	1	1	2	2	3 ↖	3	3
8	o	0	1	1	1	2	2	3	4 ↖	4
9	n	0	1	1	1	2	2	3	4	5

e c i o n

-1 ~~6~~

The explain:

I use this formula when I fill up the table:

if $x_i = y_j$

$c[i-1, j-1] + 1$

otherwise

$\max(c[i-1, j], c[i, j-1])$

After I fill up the table, I determine the length and I do like what I drew on the matrix.

∴ LCS of X and Y = ecion

Problem 3 (9 points)

8.5/9

Let $A_1, \ldots, A_4$ be matrices with dimensions 2×3, 3×10, 10×3, 3×5, respectively. In finding an optimal parenthesization of the matrix chain product $A_1 * A_2 * A_3 * A_4$, we use two tables **m[,]** and **s[,]** below. Here **m[i,j]** stores the optimal cost of computing subchain $A_i \ldots A_j$ and **s[i,j]** records the index k where the optimal parenthesization splits $A_i \ldots A_j$ between A_k and A_{k+1} for some k with $i \le k \le j-1$.

$A_1 \cdot A_2 \cdot A_3 \cdot A_4$
$2\times3 \quad 3\times10 \quad 10\times3 \quad 3\times5$

a- What is the recursive equation of **m[i,j]**?

if $i = j$, $m[i,j] = 0$,
otherwise,

$$m[i,j] = \min_{i \le k < j} \left\{ m[i,k] + m[k+1, j] + d_{i-1} * d_k * d_j \right\}$$

-0.5

b- Fill the empty entries in the two tables. Show your work in each case.

m

j \ i	1	2	3	4
4				0
3			0	
2		0		
1	0			

s

j \ i	1	2	3
4			
3			0
2		0	

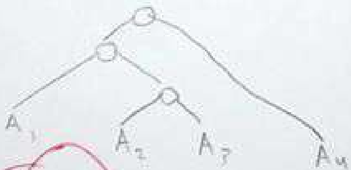
← My table in the back of Previous page.

c- Now, give the optimal parenthesization of the matrix chain product *A1*A2*A3*A4*. Show how you came up with the solution using the tables above.

$((A_1)\cdot(A_2\cdot A_3))\cdot(A_4)$

First: I compute that M[1,4]=3, so I split on A_3 (form left to right)

second: M[1,3]=1, so I split on A_1 → $(A_1)\cdot(A_2\cdot A_3)$



Problem 4 (5 points) 00/5

Consider the following equation system:

wight
value

$$\max \quad x_1 + 4x_2 + 3x_3$$

$$x_1 + 4x_2 + 3x_3 \leq 4$$

weight

$P_0 = 2 \quad P_1 = 3 \quad P_2 = 10 \quad P_3 = 3 \quad P_4 = 5$

i

S (j \ i)	1	2	3	4
1	0	60	108	138
2		0	90	135
3			0	150
4				0

(i)

(j)

M	1	2	3	4
1		1	1	3
2			2	3
3				3
4				

$$\underset{2\times3}{A_1} \cdot \underset{3\times10}{A_2} \cdot \underset{10\times3}{A_3} \cdot \underset{3\times5}{A_4}$$

* m[1,1] = 0 , m[2,2] = 0 , m[3,3] = 0 m[4,4] = 0

$\boxed{\text{* m[1,2]}}$

$A_1 \cdot A_2 = 2\times3\times10 = 60$

$\boxed{\text{* m[2,3]}}$

$A_2 \cdot A_3 = 3\times10\times3 = 90$

$\boxed{\text{* m[3,4]}}$

$A_3 \cdot A_4 = 10\times3\times5 = 150$

$\boxed{\text{* m[1,3]}}$

$(A_1 \cdot A_2) \cdot A_3$
m[1,2] + m[3,3] + 2*10*3
60 + 0 + 60 = 120

$A_1 \cdot (A_2 \cdot A_3)$
m[1,1] + m[2,3] + 2*3*3
0 + 90 + 18 = 108
minimum

$\boxed{\text{* m[2,4]}}$

$(A_2 \cdot A_3) \cdot A_4$
m[2,3] + m[4,4] + 3*3*5
90 + 0 + 45 = 135
minimum

$A_2 \cdot (A_3 \cdot A_4)$
m[2,2] + m[3,4] + 3*10*5
0 + 150 + 150 = 300

$\boxed{\text{* m[1,4]}}$

m[1,4] = min{ m[1,1] + m[2,4] + 2*3*5 , m[1,2] + m[3,4] + 2*10*5 , m[1,3] + m[4,4] + 2*3*5 }

= min{ 0 + 135 + 30 = 165 , 60 + 150 + 100 = 310 , 108 + 0 + 30 = 138 }

1	0	1	0	1	0	1		1

Problem 5 (5 points) 5/5

Give the pseudo-code of an algorithm that takes as input an array **A** of integers, and returns the length of the longest contiguous subsequence of odd numbers in **A**.

Example:

1 0 1 2 0 1 2 3 0 1 2 0

The length of the longest contiguous ~~zeros~~ odd subsequence in [1, 2, 19, 5, 4, 7, 51, 23, 22, 13, 15, 36] is 3.

What is the time complexity of your algorithm? hint: <u>Use Dynamic programming paradigm</u>.

* consider that I create a new Array called B, and the length of it is like A.

```
if( A[0]%2 != 0)
    B[0] = 1;
else
    B[0] = 0;

for (int i = 1; i < length of A ; i++){
    if( A[i]%2 != 0)
        B[i] = B[i-1]+1;
    else
        B[i] = 0;
}
```

ok! next page!

what? ~~(-1)~~

* Now I have Array B which contain of maximum number of seq. of odd numbers, so I have to create a method that find max element of array.

↓

3 2 0 4

```
int m = B[0]

for ( int i=1; i< length of B ; i++) {
    if ( B[i] > m)
        m = B[i];

}

return m;
```

which is maximum number in the array

(max seq. of odd numbers)